National College of Ireland

Configuration Manual

MSc Research Project

MSc in Cyber Security

Abhirup Bhattacharjee

Student ID: x20250029

School of Computing

National College of Ireland

Supervisor: Mr. Michael Prior

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Abhirup Bhattacharjee
Student ID:	X20250029
Programme:	MSc in Cyber Security **Year:** 2021-22
Module:	MSc Research Project
Supervisor:	Mr. Michael Prior
Submission Due Date:	15.08.2022
Project Title:	Cyber Security Intrusion Detection Deep Learning Model for Internet of Things (IoT)
Word Count:	5496 **Page Count** 22

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Abhirup Bhattacharjee

Date: 14.08.2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Abhirup Bhattacharjee
Student ID: x20250029

1 Introduction

The configuration manual plays an important role on information about all the hardware, software and procedures used in the implementation of this project. The research work employing deep learning techniques for intrusion detection in the internet of things relies heavily on the configuration manual. This guidebook includes the general setup and necessary tools for carrying out this work.

2 Hardware Requirements

- Processor: AMD Ryzen 5 5500U with Radeon Graphics 2.10 GHz
- Memory (RAM) Installed: 16 GB DDR4 3200 MHz
- System Type: Windows 10 Home, 64 Bit
- Operating System with x64-based processor
- Storage: 500 GB SSD
- GPU: 12 GB, Nvidia GeForce GTX 1650

3 Software Requirements

The IDE (Integrated Development Environment) selected for the purpose of the research was Jupyter Notebook and the programming language used was Python. Various packages of python were used for visualization and analytical purposes. The version specifications are listed below:

- Jupyter Notebook: 6.4.8
- Python 3.9.12
- Anaconda Navigator
- Microsoft Excel

As the environment for the research, Anaconda Navigator was used which is a complete package including Jupyter Notebook, Jupyter Lab, and the python setup required to execute the code. As per the system requirement, the Anaconda's 64-bit version should be installed for Windows 10. Jupyter Notebook needs to be launched from the navigator after successful installation. The lab will automatically launch on the default browser of the system, in this case the Brave Browser.

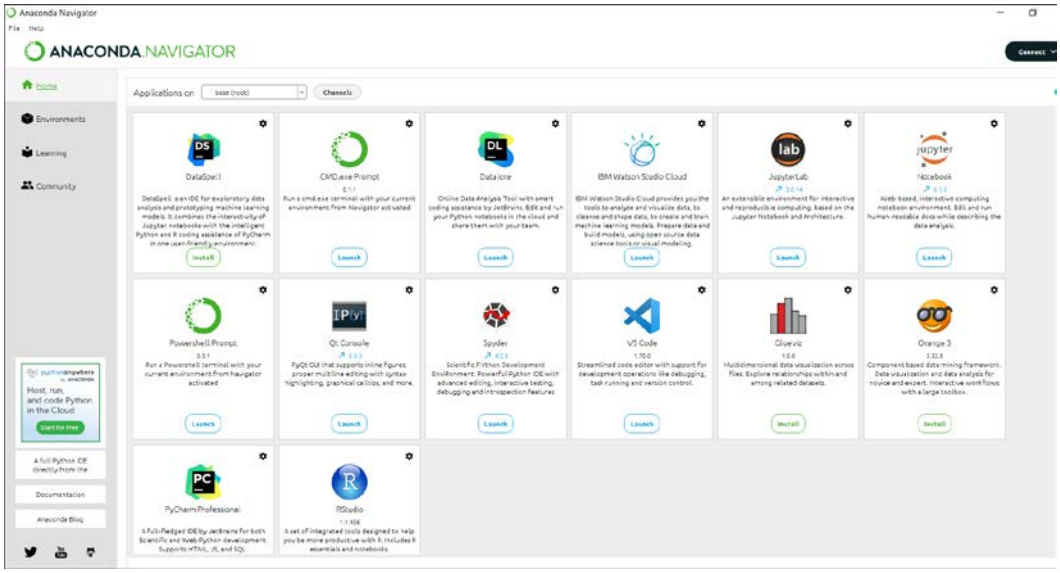

Figure 1 Anaconda and Jupyter Notebook

4 Library Package Requirements

The packages required for the research which were installed in the environment using the pip command are listed below:

- Keras – 2.9.0
- Numpy – 1.20.1
- Pandas – 1.2.4
- Matplotlib – 3.3.4

```
import sys
import keras
import sklearn
import itertools
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn import metrics
import sklearn.preprocessing
from scipy.stats import zscore
import matplotlib.pyplot as plt
from keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.utils import get_file, plot_model
from sklearn.model_selection import train_test_split
from tensorflow.keras.callbacks import EarlyStopping
from keras.layers import Dense, Dropout, Activation, Embedding
from keras.layers import LSTM, SimpleRNN, GRU, Bidirectional, BatchNormalization,Convol
from sklearn.metrics import accuracy_score, f1_score, precision_score,recall_score, con
from sklearn.model_selection import StratifiedKFold
from sklearn.linear_model import Perceptron
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, VotingClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import AdaBoostClassifier
import sklearn.metrics as metrics
from sklearn import svm, datasets
from sklearn.utils.multiclass import unique_labels
from sklearn.preprocessing import label_binarize
from sklearn.multiclass import OneVsRestClassifier
from scipy import interp
from itertools import cycle
from sklearn.datasets import make_classification
from sklearn.neighbors import KNeighborsClassifier
import warnings
warnings.filterwarnings('ignore')
from tensorflow.keras.utils import plot_model
```

Figure 2 Importing Libraries

For the purpose of the deep learning architecture, all of these libraries were imported. Here, we used a variety of library functions to generate the model. In the first line, I imported the sequence function from the keras.preprocessing library to list the sequences, and I imported the "Sequential" function from the keras.models library to start the CNN model. Dense, Dropout, Activation, and Lambda functions from the keras.layers library were imported because the CNN layer will require all four of these.

5 Dataset Description and Data Pre-processing

```
In [2]: df = pd.read_csv('NSL-KDD/KDDTrain+.txt', header=None)
        df.head()
```

Out[2]:

	0	1	2	3	4	5	6	7	8	9	...	33	34	35	36	37	38	39	40	41	42
0	0	tcp	ftp_data	SF	491	0	0	0	0	0	...	0.17	0.03	0.17	0.00	0.00	0.00	0.05	0.00	normal	20
1	0	udp	other	SF	146	0	0	0	0	0	...	0.00	0.60	0.88	0.00	0.00	0.00	0.00	0.00	normal	15
2	0	tcp	private	S0	0	0	0	0	0	0	...	0.10	0.05	0.00	0.00	1.00	1.00	0.00	0.00	neptune	19
3	0	tcp	http	SF	232	8153	0	0	0	0	...	1.00	0.00	0.03	0.04	0.03	0.01	0.00	0.01	normal	21
4	0	tcp	http	SF	199	420	0	0	0	0	...	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	normal	21

5 rows × 43 columns

```
In [3]: qp = pd.read_csv('NSL-KDD/KDDTest+.txt', header=None)
        qp.head()
```

Out[3]:

	0	1	2	3	4	5	6	7	8	9	...	33	34	35	36	37	38	39	40	41	42
0	0	tcp	private	REJ	0	0	0	0	0	0	...	0.04	0.06	0.00	0.00	0.0	0.0	1.00	1.00	neptune	21
1	0	tcp	private	REJ	0	0	0	0	0	0	...	0.00	0.06	0.00	0.00	0.0	0.0	1.00	1.00	neptune	21
2	2	tcp	ftp_data	SF	12983	0	0	0	0	0	...	0.61	0.04	0.61	0.02	0.0	0.0	0.00	0.00	normal	21
3	0	icmp	eco_i	SF	20	0	0	0	0	0	...	1.00	0.00	1.00	0.28	0.0	0.0	0.00	0.00	saint	15
4	1	tcp	telnet	RSTO	0	15	0	0	0	0	...	0.31	0.17	0.03	0.02	0.0	0.0	0.83	0.71	mscan	11

5 rows × 43 columns

Figure 3 Importing Dataset

```
In [44]: oos_pred = []
         k_dict = {'accuracy' : [], 'detectionRate' : [], 'falsepositiverate' : [], 'confusionmatrix' : []}

In [45]: for k in range(2,11,2):
             k_dict['accuracy'].append([])
             k_dict['confusionmatrix'].append([])
             kfold = StratifiedKFold(n_splits=k,shuffle=True,random_state=42)
             kfold.get_n_splits(combined_data_X,y_train)
             for train_index, test_index in kfold.split(combined_data_X,y_train):
                 train_X, test_X = combined_data_X.iloc[train_index], combined_data_X.iloc[test_index]
                 train_y, test_y = y_train.iloc[train_index], y_train.iloc[test_index]


                 x_columns_train = new_train_df.columns.drop('Class')
                 x_train_array = train_X[x_columns_train].values
                 x_train_1=np.reshape(x_train_array, (x_train_array.shape[0], x_train_array.shape[1], 1))

                 dummies = pd.get_dummies(train_y)
                 outcomes = dummies.columns
                 num_classes = len(outcomes)
                 y_train_1 = dummies.values

                 x_columns_test = new_train_df.columns.drop('Class')
                 x_test_array = test_X[x_columns_test].values
                 x_test_2=np.reshape(x_test_array, (x_test_array.shape[0], x_test_array.shape[1], 1))

                 dummies_test = pd.get_dummies(test_y)
                 outcomes_test = dummies_test.columns
                 num_classes = len(outcomes_test)
                 y_test_2 = dummies_test.values


                 model.fit(x_train_1, y_train_1,validation_data=(x_test_2,y_test_2), epochs=1)
                 #cnn.fit(x_train_1, y_train_1,validation_data=(x_test_2,y_test_2), epochs=1)
                 pred = model.predict(x_test_2)
                 pred = np.argmax(pred,axis=1)
                 y_eval = np.argmax(y_test_2,axis=1)
                 score = metrics.accuracy_score(y_eval, pred)
                 k_dict['accuracy'][-1].append(score)
                 cm = confusion_matrix(y_eval, pred, labels=[0,1,2,3,4])
                 k_dict['confusionmatrix'][-1].append(cm)
                 print("Validation score: {}".format(score))
```

Figure 4 Data Pre-processing

6 Directory Structure and Execution

The dataset needs to be downloaded and saved in “NSL-KDD” folder. The FYP_IDS_NSL.ipnyb file and CNN-DNN.ipnyb file needs to be executed.

6.1 Model Training Summary

```
In [18]: def build_network():

             models = []
             model = Sequential()
             model.add(Dense(64, input_dim=122))

             model.add(Dense(32))
             model.add(Activation('relu'))
             model.add(Dropout(.15))
             model.add(Dense(32))
             model.add(Activation('relu'))
             model.add(Dropout(.15))
             model.add(Dense(32))
             model.add(Activation('relu'))
             model.add(Dropout(.15))
             model.add(Dense(5))
             model.add(Activation('softmax'))

             model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

             return model


         def build_cnn_network():
             model = Sequential()
             model.add(Conv1D(32, kernel_size=5, strides=1,
                         activation='relu',
                         input_shape=(122,1)))
             model.add(MaxPooling1D(pool_size=2, strides=2))
             model.add(Conv1D(64, 5, activation='relu'))
             model.add(MaxPooling1D(pool_size=2))
             model.add(Flatten())
             model.add(Dense(1000, activation='relu'))
             model.add(Dense(5, activation='softmax'))
             model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
             return model
```

```
#AdaBoost with decision tree
decision_clf = DecisionTreeClassifier()
clf = AdaBoostClassifier(decision_clf)
clf.fit(X_train,Y_train)
train_acc = clf.score(X_train, Y_train)
test_acc = clf.score(X_test, Y_test)
y_pred = clf.predict(X_test)
print("Training accuracy is:", train_acc )
print("Testing accuracy is:", test_acc)
```

```
Training accuracy is: 0.999833297611393
Testing accuracy is: 0.7649041873669269
```

```
#Logistic Regression
clf = LogisticRegression (solver='liblinear', multi_class='auto')
clf.fit(X_train, Y_train)
train_acc = clf.score(X_train, Y_train)
test_acc = clf.score(X_test, Y_test)
y_pred = clf.predict(X_test)
print("Training accuracy is:", train_acc )
print("Testing accuracy is:", test_acc)
```

```
Training accuracy is: 0.9762488787279814
Testing accuracy is: 0.7498669268985095
```

```
In [70]: #Random Forest
         clf = RandomForestClassifier()
         clf.fit(X_train,Y_train)
         train_acc = clf.score(X_train, Y_train)
         test_acc = clf.score(X_test, Y_test)
         y_pred = clf.predict(X_test)
         print("Training accuracy is:", train_acc )
         print("Testing accuracy is:", test_acc)

         Training accuracy is: 0.9998412358203742
         Testing accuracy is: 0.7588715400993612
```

```
In [28]: from sklearn.metrics import f1_score, precision_score, recall_score, accuracy_score
         print('Precision: %.3f' % f1_score(true_lbls_dnn, pred_lbls_dnn, average='weighted'))
         print('Precision: %.3f' % precision_score(true_lbls_dnn, pred_lbls_dnn,average='weighted'))
         print('Recall: %.3f' % recall_score(true_lbls_dnn, pred_lbls_dnn, average='weighted'))
         print('Accuracy: %.3f' % accuracy_score(true_lbls_dnn, pred_lbls_dnn))

         Precision: 0.756
         Precision: 0.824
         Recall: 0.789
         Accuracy: 0.789

In [29]: print('Precision: %.3f' % f1_score(true_lbls_cnn, pred_lbls_cnn, average='weighted'))
         print('Precision: %.3f' % precision_score(true_lbls_cnn, pred_lbls_cnn,average='weighted'))
         print('Recall: %.3f' % recall_score(true_lbls_cnn, pred_lbls_cnn, average='weighted'))
         print('Accuracy: %.3f' % accuracy_score(true_lbls_cnn, pred_lbls_cnn))

         Precision: 0.725
         Precision: 0.821
         Recall: 0.766
         Accuracy: 0.766
```

Figure 5 Model Accuracy Summary